PERGAMON

Computers & Fluids 30 (2001) 917–925

computers & fluids

www.elsevier.com/locate/compfluid

Iterative modified approximate factorization

Robert W. MacCormack *

Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305-4035, USA

Accepted 1 March 2001

Abstract

An implicit procedure for minimizing the error introduced by approximate factorization in solving scalar or block multidiagonal matrix equations is presented with applications to fluid dynamics. High convergence rates of approximately 0.8 are achieved upon application to a wide range of problems. The implicit procedure can be combined with a multigrid procedure for further convergence acceleration.

1. Introduction

Implicit numerical methods for solving pentadiagonal, or higher, matrix equations, as often found in solution procedures for fluid dynamics problems, replace by approximate factorization (AF) the original matrix with a sequence of efficiently invertible matrix factors, each usually tridiagonal. This procedure introduces errors because the resulting matrix, upon remultiplication, contains different matrix elements than the original matrix. The errors cause a reduction in convergence speed and the necessity of using time steps much smaller than that required to follow the time evolution of unsteady flows, particularly viscous flows or those converging to steady state solutions. Nevertheless, this procedure, called standard AF herein, developed by Beam and Warming [1] and Briley and McDonald [2] for solving the equations of the fluid dynamics, has been the work horse of computational fluid dynamics since the late 1970's.

A modified approximate factorization (MAF) procedure, also called diagonally dominant alternate direction implicit (DDADI), that can regain a significant part of the convergence rate loss

* Tel.: +1-650-723-4627.

0045-7930/01/$ - see front matter
PII: S0045-7930(01)00035-4

caused by the standard AF and often operate with unlimited time step size was presented in the early 1960's in Ref. [3] and exploited by Bardina and Lombard [4] in 1987. However, this modification made little headway into use in computational fluid dynamics procedures. An analysis by MacCormack and Pulliam [5] showed that this procedure is not always stable unless a relaxation parameter (given later in this paper by α) is introduced and over relaxed. Finally, an additional improvement, an iterative DDADI procedure [6] called MAF(k) herein, that can restore all the lost convergence speed caused by standard AF is presented in the following.

2. Governing equations

Consider the unsteady equations of compressible viscous flow in two dimensions represented in general coordinates ξ and η as follows:

$$\frac{\partial U}{\partial t}+\frac{\partial F}{\partial \xi}+\frac{\partial G}{\partial \eta}=0. \tag{1}$$

A finite difference/volume equation, written in *delta* law form, approximating the above equation, is given by Eq. (2) below

$$\left\{I+\Delta t\left(\frac{D_{-}\cdot}{\Delta \xi}\overline{A}^{n}_{i+1/2,j}+\frac{D_{-}\cdot}{\Delta \eta}\overline{B}^{n}_{i,j+1/2}\right)\right\}\delta U^{n+1}_{i,j}=-\Delta t\left(\frac{F^{n}_{i+1/2,j}-F^{n}_{i-1/2,j}}{\Delta \xi}+\frac{G^{n}_{i,j+1/2}-G^{n}_{i,j-1/2}}{\Delta \eta}\right) \tag{2}$$

where $\overline{A}$ and $\overline{B}$ are the matrix Jacobians of the flux vectors F and G with respect to state vector U, respectively, and the dots, $\cdot$, appearing in the equation indicate that the difference operators operate on all factors to the right. Upon application of the difference operators an equation of the following form is usually obtained:

$$\mathbf{B}_{i,j}\delta U^{n+1}_{i,j+1}+\mathbf{A}_{i,j}\delta U^{n+1}_{i,j}+\mathbf{C}_{i,j}\delta U^{n+1}_{i,j-1}+\mathbf{D}_{i,j}\delta U^{n+1}_{i+1,j}+\mathbf{E}_{i,j}\delta U^{n+1}_{i-1,j}=\Delta U^{n}_{i,j} \tag{3}$$

where $\mathbf{A}_{i,j}$, $\mathbf{B}_{i,j}$, $\mathbf{C}_{i,j}$, $\mathbf{D}_{i,j}$ and $\mathbf{E}_{i,j}$ are block matrix elements and

$$\Delta U^{n}_{i,j}=-\Delta t\left(\frac{F^{n}_{i+1/2,j}-F^{n}_{i-1/2,j}}{\Delta \xi}+\frac{G^{n}_{i,j+1/2}-G^{n}_{i,j-1/2}}{\Delta \eta}\right)$$

Eq. (3) can also be expressed in full matrix form:

$$\mathbf{M}\ \cdot\ [\delta \mathbf{U}]=[\Delta \mathbf{U}]$$

where

$$\mathbf{M} = \begin{bmatrix}
x & x & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
x & x & x & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & x & x & x & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & x & x & x & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot \\
x & \cdot & \cdot & x & x & x & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot \\
\cdot & x & \cdot & \cdot & x & x & x & \cdot & \cdot & x & \cdot & \cdot & \cdot \\
\cdot & \cdot & \overline{\mathbf{D}} & \cdot & \cdot & \overline{\mathbf{B}} & \overline{\mathbf{A}} & \overline{\mathbf{C}} & \cdot & \cdot & \overline{\mathbf{E}} & \cdot & \cdot \\
\cdot & \cdot & \cdot & x & \cdot & \cdot & x & x & x & \cdot & \cdot & x & \cdot \\
\cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & x & x & x & \cdot & \cdot & x \\
\cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & x & x & x & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & x & x & x & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & x & x & x \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & x & x
\end{bmatrix}.$$

Matrix $\mathbf{M}$ is of size $[(n \times n) \times (n \times n)]$ and vectors $[\delta\mathbf{U}]$ and $[\Delta\mathbf{U}]$ are of length $(n \times n)$ on an $(n \times n)$ mesh.

The block element matrices $\overline{\mathbf{A}}_{i,j}$, $\overline{\mathbf{B}}_{i,j}$, $\overline{\mathbf{C}}_{i,j}$, $\overline{\mathbf{D}}_{i,j}$ and $\overline{\mathbf{E}}_{i,j}$ are defined below with $\alpha = 1.0$

$$\overline{\mathbf{A}}_{i,j} = I + \alpha\frac{\Delta t}{\Delta x}\left(\overline{A}^{n}_{+i+1/2,j} - \overline{A}^{n}_{-i-1/2,j}\right) + \alpha\frac{\Delta t}{\Delta y}\left(\overline{B}^{n}_{+i,j+1/2} - \overline{B}^{n}_{-i,j-1/2}\right)$$

$$\overline{\mathbf{B}}_{i,j} = +\alpha\frac{\Delta t}{\Delta y}\overline{B}^{n}_{-i,j+1/2} \quad \overline{\mathbf{C}}_{i,j} = -\alpha\frac{\Delta t}{\Delta y}\overline{B}^{n}_{+i,j-1/2} \quad \overline{\mathbf{D}}_{i,j} = +\alpha\frac{\Delta t}{\Delta x}\overline{A}^{n}_{-i+1/2,j} \quad \overline{\mathbf{E}}_{i,j} = -\alpha\frac{\Delta t}{\Delta x}\overline{A}^{n}_{+i-1/2,j}. \tag{4}$$

The *bars* appearing on the matrix elements indicate that the modified-Steger–Warming method is being used to approximate these Jacobian matrices using arithmetically averaged data from both sides of the flux surface. The Roe method, using geometrically averaged data, could be used as well. The subscripts appearing on the Jacobian matrices, for example $\overline{A}^{n}_{+i-1/2,j}$, indicate the sign of the split matrix and the flux surface location. For this example, the split Jacobian matrix contains only nonnegative eigenvalues and the flux surface lies midway between mesh points (i, j) and $(i + 1, j)$.

3. Approximate factorization

The standard approach to solve Eq. (2) is AF, which when applied yields

$$\left\{I + \Delta t\frac{D_{-\cdot}}{\Delta\xi}\overline{A}^{n}_{i+1/2,j}\right\}\left\{I + \frac{D_{-\cdot}}{\Delta\eta}\overline{B}^{n}_{i,j+1/2}\right\}\delta U^{n+1}_{i,j} = -\Delta U_{i,j}.$$

The matrix $\mathbf{M}$ is factored by

$$\mathbf{M} \simeq \mathbf{M}_{\xi} \cdot \mathbf{M}_{\eta}. \tag{5}$$

These matrix factors are of form

$$\mathbf{M}_\xi = \begin{bmatrix} x & \cdot & x & \cdot & \cdot & \cdot & \cdot \\ \cdot & x & \cdot & x & \cdot & \cdot & \cdot \\ x & \cdot & x & \cdot & x & \cdot & \cdot \\ \cdot & \overline{\mathbf{D}} & \cdot & \overline{\mathbf{A}}_\xi & \cdot & \overline{\mathbf{E}} & \cdot \\ \cdot & \cdot & x & \cdot & x & \cdot & x \\ \cdot & \cdot & \cdot & x & \cdot & x & \cdot \\ \cdot & \cdot & \cdot & \cdot & x & \cdot & x \end{bmatrix}$$

and

$$\mathbf{M}_\eta = \begin{bmatrix} x & x & \cdot & \cdot & \cdot & \cdot & \cdot \\ x & x & x & \cdot & \cdot & \cdot & \cdot \\ \cdot & x & x & x & \cdot & \cdot & \cdot \\ \cdot & \cdot & \overline{\mathbf{B}} & \overline{\mathbf{A}}_\eta & \overline{\mathbf{C}} & \cdot & \cdot \\ \cdot & \cdot & \cdot & x & x & x & \cdot \\ \cdot & \cdot & \cdot & \cdot & x & x & x \\ \cdot & \cdot & \cdot & \cdot & \cdot & x & x \end{bmatrix}.$$

Each factor represents a block tridiagonal matrix. During matrix inversion, each need to be of only dimension $(n \times n)$ when solved *line by line* through each mesh direction. The matrix elements $\overline{\mathbf{B}}_{i,j}$, $\overline{\mathbf{C}}_{i,j}$, $\overline{\mathbf{D}}_{i,j}$ and $\overline{\mathbf{E}}_{i,j}$ are defined as before in Eq. (4) and

$$\overline{\mathbf{A}}_{\xi i,j} = I + \frac{\Delta t}{\Delta x}\left(\overline{A}^n_{+i+1/2,j} - \overline{A}^n_{-i-1/2,j}\right)$$

$$\overline{\mathbf{A}}_{\eta i,j} = I + \frac{\Delta t}{\Delta x}\left(\overline{B}^n_{+i,j+1/2} - \overline{B}^n_{-i,j-1/2}\right).$$

Unfortunately, upon matrix multiplication of the two factors in Eq. (5), none of the original matrix elements are returned exactly and some *zero* elements of **M** become significantly *nonzero*. This factorization error reduces accuracy and slows algorithm convergence.

4. Modified approximate factorization

The AF procedure can be modified to significantly reduce the adverse effects of decomposition error. The modified procedure has the property of Stone's strongly implicit method (SIP) [7] for matrix decomposition of returning exactly the original *nonzero* elements of the matrix **M** upon factor multiplication, although some originally *zero* elements become *nonzero*. This modification was used by Bardina and Lombard [4] in 1987, which they called their DDADI procedure. The basic idea of the diagonally dominant procedure also appears much earlier in the literature. It is apparently rediscovered every decade or so and then, unfortunately, forgotten by the computational community. When applied to the matrix **M**

$$\mathbf{M} \simeq \mathbf{M}'_\xi \cdot [\mathbf{D}]^{-1} \cdot \mathbf{M}'_\eta \tag{6}$$

with

$$\mathbf{M}'_{\xi} = \begin{bmatrix} x & \cdot & x & \cdot & \cdot & \cdot & \cdot \\ \cdot & x & \cdot & x & \cdot & \cdot & \cdot \\ x & \cdot & x & \cdot & x & \cdot & \cdot \\ \cdot & \overline{\mathbf{D}} & \cdot & \overline{\mathbf{A}} & \cdot & \overline{\mathbf{E}} & \cdot \\ \cdot & \cdot & x & \cdot & x & \cdot & x \\ \cdot & \cdot & \cdot & x & \cdot & x & \cdot \\ \cdot & \cdot & \cdot & \cdot & x & \cdot & x \end{bmatrix}$$

$$\mathbf{M}'_{\eta} = \begin{bmatrix} x & x & \cdot & \cdot & \cdot & \cdot & \cdot \\ x & x & x & \cdot & \cdot & \cdot & \cdot \\ \cdot & x & x & x & \cdot & \cdot & \cdot \\ \cdot & \cdot & \overline{\mathbf{B}} & \overline{\mathbf{A}} & \overline{\mathbf{C}} & \cdot & \cdot \\ \cdot & \cdot & \cdot & x & x & x & \cdot \\ \cdot & \cdot & \cdot & \cdot & x & x & x \\ \cdot & \cdot & \cdot & \cdot & \cdot & x & x \end{bmatrix}$$

and diagonal matrix

$$\mathbf{D} = \begin{bmatrix} x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \overline{\mathbf{A}} & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x \end{bmatrix}.$$

The block matrix elements appearing above are again defined by Eq. (4) with the exception that the parameter α has now been set to 2.

In three dimensions $\mathbf{M}$ can be decompositioned by

$$\mathbf{M} \simeq \mathbf{M}'_{\xi} \cdot [\mathbf{D}]^{-1} \cdot \mathbf{M}'_{\eta} \cdot [\mathbf{D}]^{-1} \cdot \mathbf{M}'_{\zeta}.$$

Although the original *nonzero* elements of matrix $\mathbf{M}$ are returned exactly by the above MAF decomposition procedure, other formerly *zero* elements are disturbed and can contribute to both error and reduced convergence speed. The following section describes an iterative procedure for eliminating or further reducing this remaining decomposition error.

5. Removal of decomposition error

Eq. (6) can be expanded as follows:

$$\mathbf{M} \simeq \mathbf{M}'_{\xi} \cdot [\mathbf{D}]^{-1} \cdot \mathbf{M}'_{\eta} = \mathbf{M} + \mathbf{P}.$$

The equation actually solved, instead of Eq. (3), is

$$\mathbf{M} \cdot [\delta \mathbf{U}] + \mathbf{P} \cdot [\delta \mathbf{U}] = [\Delta \mathbf{U}]. \tag{7}$$

The difference between Eqs. (3) and (7) is the decomposition error term $\mathbf{P}\cdot[\delta\mathbf{U}]$. We now present an iterative procedure for removing this decomposition error.

The decomposition error term is fed back into the matrix equation on the right hand side for self cancellation. A k-step iterative MAF algorithm [6], MAF(k), is defined as follows:

MAF(k) algorithm

$$\mathbf{M}\cdot[\delta\mathbf{U}^{(k)}]+\mathbf{P}\cdot[\delta\mathbf{U}^{(k)}]=[\Delta\mathbf{U}]+\mathbf{P}\cdot[\delta\mathbf{U}^{(k-1)}]$$

where $[\delta\mathbf{U}^{(0)}]=[0]$, $k=1,2,3,\ldots$. MAF(1) is the same as Eq. (7) above. For this iterative procedure to work, each MAF iteration must be numerically stable and the sequence must converge. For it to be efficient, the number of iterations must be kept small, ideally at two. A stability analysis and examples showing sequence convergence have been given [6]. The optimum value for the maximum number of k sub-iterations per time step was shown to be two for flows going to steady states.

Implementation of the MAF(k) algorithm: The block matrix elements of $\mathbf{M}$ need to be calculated, once or as often as needed if not saved, to form the matrices $\mathbf{M}'_\xi$, $\mathbf{D}$, $\mathbf{M}'_\eta$. The matrix P need never be computed.

The algorithm can be also written as, for $k=1,2,\ldots$

$$\mathbf{M}'_\xi\cdot[\mathbf{D}]^{-1}\cdot\mathbf{M}'_\eta\cdot[\delta\mathbf{U}^{(k)}]=[\Delta\mathbf{U}]+[\mathbf{R}^k]$$

where $[\mathbf{R}^1]=[0]$, and for $k>1$

$$[\mathbf{R}^k]=[\Delta\mathbf{U}]-\mathbf{M}\cdot[\delta\mathbf{U}^{(k-1)}]+[\mathbf{R}^{k-1}].$$

Alternatively, the algorithm can be rewritten, without the need to define either $\mathbf{P}$ or $\mathbf{R}^k$, as

$$\mathbf{M}'_\xi\cdot[\mathbf{D}]^{-1}\cdot\mathbf{M}'_\eta\cdot[\delta\mathbf{U}^{(k)}-\delta\mathbf{U}^{(k-1)}]=[\Delta\mathbf{U}]-\mathbf{M}\cdot[\delta\mathbf{U}^{(k-1)}].$$

6. Added numerical dissipation

Numerical dissipation is the fundamental requirement for numerical stability. It has to be present to obtain numerical solutions. It is introduced by the choices made in discretization. When this is insufficient, more must be added. This section presents a rational way to add numerical dissipation.

In fluid dynamics applications, one would like to preserve the physical balance of the Navier–Stokes stress tensor, which is frame independent. It is hopeless to add artificial viscosity in multidimensional curvilinear coordinate systems and simultaneously preserve this physical balance – with few exceptions.

One exception is to use the already frame independent Navier–Stokes viscous stress tensor itself as a vehicle to add numerical dissipation. Similar to the concept of an eddy viscosity used to include the effects of turbulent mixing in the Reynolds averaged Navier–Stokes equations, the addition of numerical viscosity by augmenting the natural viscosity can be used in the frame independent Navier–Stokes stress tensor, even when solving the Euler equations, to control numerical instability.

$$\mu \leftarrow \mu_{\text{physical}} + \mu_{\text{numerical}}. \tag{8}$$

Numerical difficulties arise at the foot of shock waves and in regions where oscillation occurs in the convection velocity within subsonic flow.

(1) Along the shock wave, the numerical viscosity can be chosen to be

$$\mu_{\text{numerical}_i} = 1/2 \cdot \Delta h_{\xi i} \cdot \rho_i c_i \tag{9}$$

where Δh_ξ represents the grid spacing distance in the ξ direction, assumed crossing through the shock, and the ρ is the density. At both points adjacent to the shock the viscosity is augmented as shown above in Eq. (8).

(2) Within subsonic regions with oscillation in convection speed q_n.

$$\mu_{\text{numerical}_i} = 0.05 \cdot \min\{\Delta h_\xi, \Delta h_\eta, \Delta h_\zeta\} \cdot (\rho_1 |q_{n1}| + \rho_2 |q_{n2}|) \cdot \frac{|q_{n3} - 3(q_{n1} - q_{n2}) - q_{n4}|}{|q_{n3}| + 3(|q_{n1}| + |q_{n2}|) + |q_{n4}| + \epsilon} \tag{10}$$

where points 1, 2, 3 and 4 are adjacent points about grid $i =$ "1" and $\epsilon \simeq 10^{-9}$ is added to prevent division by zero in motionless regions. The quotient factor is bounded by one and is proportional to $1/8 q_n \cdot \Delta\xi^3 \cdot |\partial^3 q_n / \partial \xi^3|$. The above value of numerical viscosity is added, as in Eq. (8), at points i (1) and $i + 1$ (2).

7. Computational results and conclusion

The method just described was successfully applied to several aerodynamic flows [5,8], ranging from low subsonic to hypersonic Mach numbers – (1). The Euler equations at Mach 0.2 for flow past an ellipse (Fig. 1) (2) the Reynolds averaged Navier–Stokes equations for transonic flow past an airfoil (Fig. 2) (3) the Navier–Stokes equations for Mach 5 flow past a sphere (Fig. 3) and real gas equilibrium flow past a sphere at Mach 18 (Fig. 4). Convergence rates of 0.8, CFL numbers of one million and computational times one-fifth that for standard AF were achieved.

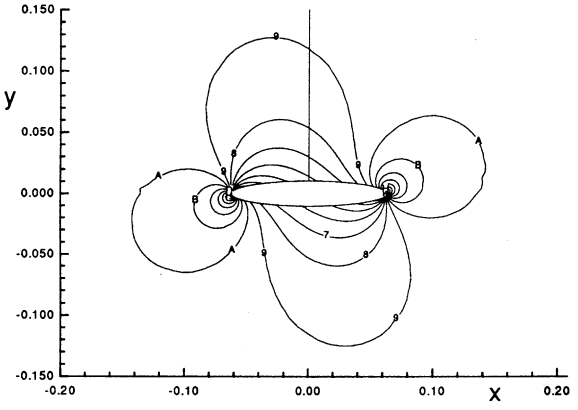

Fig. 1. Pressure contours for subsonic flow about an ellipse.

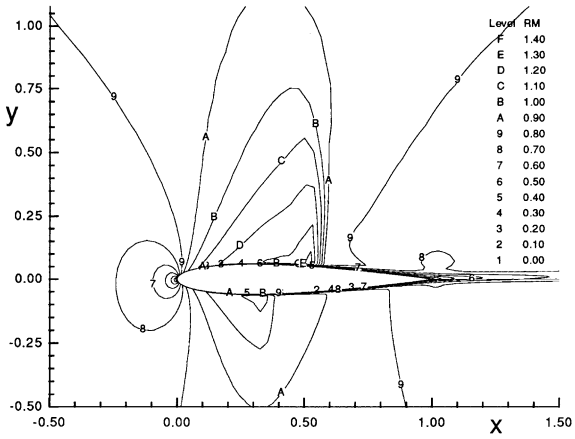

Fig. 2. Mach contours for transonic flow past an airfoil.

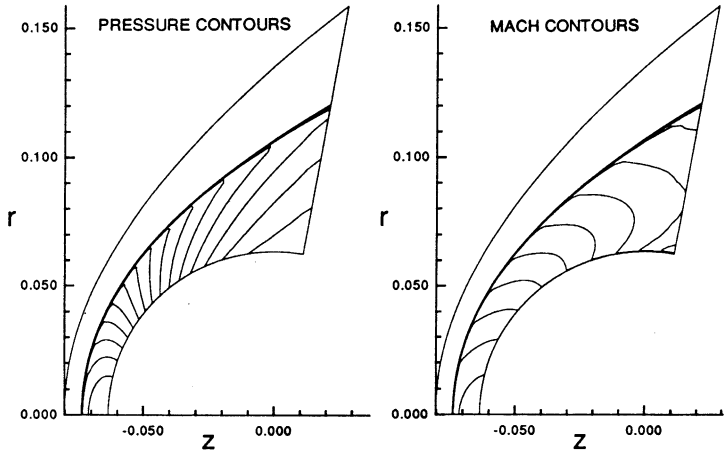

Fig. 3. Pressure and Mach contours for supersonic flow past a sphere.

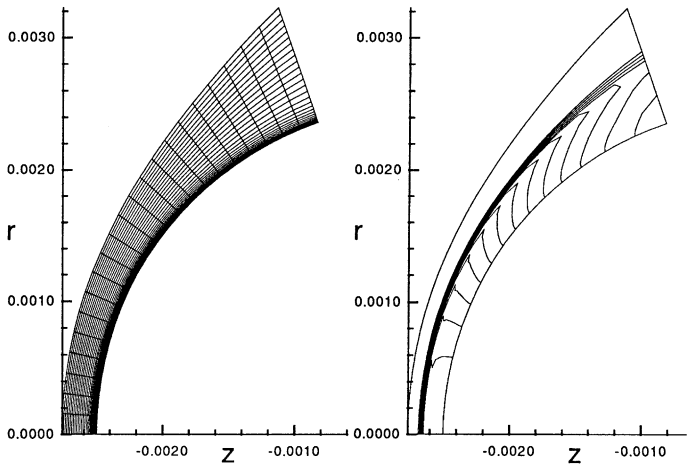

Fig. 4. Mesh and pressure contours for Mach 18 real gas flow past a sphere.

References

[1] Beam R, Warming RF. An implicit factored scheme for the compressible Navier–Stokes equation. AIAA J 1978;16:293–402.
[2] Briley WR, McDonald H. Solution of the multidimensional compressible Navier–Stokes equations by a generalized implicit method. J Computat Phys 1977;24:372–97.
[3] Varga RS. Matrix iterative analysis. Englewood Cliff, NJ: Prentice Hall; 1962.
[4] Bardina J, Lombard CK. Three dimensional hypersonic flow simulations with the CSCM implicit upwind Navier–Stokes method. AIAA Paper No. 87–1114, 1987.
[5] MacCormack RW, Puillam T. Assessment of a new numerical procedure for fluid dynamics. AIAA Paper No. 98–2821, 1998.
[6] MacCormack RW. A new implicit algorithm for fluid flow. AIAA Paper No. 97–2100, 1997.
[7] Stone HL. Iterative solution of implicit approximations of multidimensional partial differential equations. Siam J Numer Anal 1968;5(3):530–58.
[8] MacCormack RW. A fast and accurate method for solving the Navier–Stokes equations. 21st ICAS Congress, Melbourne, Australia, 13–18 September, 1998.